J. Symbolic Computation (1987) **3**, 217–229

An Algebraic Approach to Unification Under Associativity and Commutativity

ALBRECHT FORTENBACHER

Institut für Informatik I, Universität Karlsruhe,
D–7500 Karlsruhe, Federal Republic of Germany

(*Received* 20 *May*, 1985)

From the work of Siekmann & Livesey, and Stickel it is known how to unify two terms in an associative and commutative theory: transfer the terms into Abelian strings, look for mappings which solve the problem in the Abelian monoid, and decide whether a mapping can be regarded as a unifier. Very often most of the mappings are thus eliminated, and so it is crucial for efficiency either to not create these unnecessary solutions or to remove them as soon as possible. The following work formalises the transformations between the free algebra and this monoid. This leads to an algorithm which uses maximal information for its search for solutions in the monoid. It is both very efficient and easily verifiable. Some applications of this algorithm are shown in the appendix.

Introduction

Term rewriting systems are unable to handle mathematical theories with associativity and commutativity in the ordinary way. The usual approach of ordering the equations and using them as rewrite rules does not work. A rule of the form $x \circ y \to y \circ x$ will make such a system non-Noetherian. One possible solution to this problem is to build associativity and commutativity into the matching and unification algorithms.

Siekmann & Livesey (1976) and Stickel (1975, 1981) independently presented algorithms to unify two terms with an associative and commutative operator. Both algorithms flatten terms and regard the argument lists as Abelian strings (multisets) thus transferring a unification problem in the free algebra into a problem in an Abelian monoid. The algorithms differ in the way in which they find mappings which equate strings. Siekmann & Livesey restrict themselves to strings over variables and constants. This allows them to derive a system of diophantine equations and use the solutions to generate the set of unifiers directly. Their algorithm is very efficient, but did not seem to be applicable to terms of a more general form, such as terms containing more than one operator. Recently, Herold & Siekmann (1985) presented an algorithm which solves this deficiency.

Stickel uses a "variable abstraction", which reduces the set of equations to a singleton. With every mapping derived from the solutions to this equation he gets a set of pairs of terms to which the algorithm is applied recursively. If this fails, the mapping has to be disregarded.

The following work describes and formalises the connections between solutions of a

0747–7171/87/030217+13 $03.00/0

diophantine equation and a unifier for a pair of terms. An algorithm directed by this information selects only a few combinations of solutions which are expected to gain a unifier, and reduces the number of recursive unification calls rapidly.

In section 1 we show how to construct an endomorphism which solves a unification problem in an Abelian monoid. Next we formalise the constraints on endomorphisms serving as substitutions. Then a unification algorithm is presented, together with proofs of its termination, correctness, and completeness. Finally, in section 4 we give some hints on how to efficiently implement the algorithm. The reader is assumed to be familiar with the basics of Σ-algebras and term rewriting systems. Given a signature Σ and a set of variables V, the free algebra $T(\Sigma \cup V)$ is the algebra of all terms over Σ and V. A substitution is a Σ-endomorphism on $T(\Sigma \cup V)$. And given a set of equations, E, the set of all associative and commutative operators Σ_{ac} consists of all $\circ \in \Sigma$ with both $x \circ y = y \circ x \in E$ and $x \circ (y \circ z) = (x \circ y) \circ z \in E$. For details, see the survey by Huet & Oppen (1980).

1. An Algebraic Problem

In this section we solve a unification problem in a free Abelian monoid. Let h_1 and h_2 be two elements of an Abelian monoid $(H, \circ)$ with identity ε and $h_1, h_2 \in H - \{\varepsilon\}$, and regard "=" as the finest congruence relation defined by associativity and commutativity of "$\circ$". The problem we solve is that of finding a "$\circ$"-endomorphism ϕ with $\phi(h_1) = \phi(h_2)$.

TERMINOLOGY. We "normalise" h_1 and h_2 by removing common arguments and grouping the remaining ones:

$$\left.\begin{aligned} h_1 &= a_1^{p_1} \circ \ldots \circ a_m^{p_m} \\ h_2 &= a_{m+1}^{p_{m+1}} \circ \ldots \circ a_n^{p_n} \end{aligned}\right\} \text{with } a_i \in H - \{\varepsilon\} \text{ and } a_{i_1} \neq a_{i_2} \text{ for } i_1 \neq i_2.$$

Now all a_i are pairwise distinct, and no a_i can be represented as $b_1 \circ b_2$ with $b_1, b_2 \in H - \{\varepsilon\}$.

DEFINITION. For h_1, h_2 as above we define:

- a diophantine equation E_{h_1, h_2}: $\sum_{i=1}^{m} p_i x_i - \sum_{i=m+1}^{n} p_i x_i = 0$.
- S_{h_1, h_2}: $\{s \in N^n | s \text{ is a solution of } E_{h_1, h_2}\}$.
- B_{h_1, h_2}: $\{s \in S_{h_1, h_2} | s \text{ is a basic solution of } E_{h_1, h_2}\}$
  (a solution is basic if it is neither trivial nor the sum of two non-trivial solutions).

These definitions now enable us to construct an endomorphism ϕ under which h_1 and h_2 are equated. The method is as follows:

(1) Choose $s_1, \ldots, s_r \in S_{h_1, h_2}$ and $k_1, \ldots, k_r \in H - \{\varepsilon\}$ with $r \geqslant 1$.
(Let s_j be $(s_{j1}, \ldots, s_{jn})$ for $1 \leqslant j \leqslant r$).
(2) Define ϕ on $a_1, \ldots, a_n$:

$$\phi(a_i) = k_1^{s_{1i}} \circ \ldots \circ k_r^{s_{ri}} \quad \text{for } 1 \leqslant i \leqslant n.$$

LEMMA 1. $\phi(h_1) = \phi(h_2)$.

PROOF

- $\phi(h_1)$ and $\phi(h_2)$ can be represented as $k_1^{m_1} \circ \ldots \circ k_r^{m_r}$ and $k_1^{n_1} \circ \ldots \circ k_r^{n_r}$ where all k_j are pairwise distinct (see the definition of ϕ in step (2)).
- It remains to prove: $\forall\ 1 \leqslant j \leqslant r\ [m_j = n_j]$. This follows from

$$m_j = \sum_{i=1}^{m} p_i s_{ji}, \qquad n_j = \sum_{i=m+1}^{n} p_i s_{ji}$$

and $(s_{j1}, \ldots, s_{jn})$ is a solution of E_{h_1, h_2}. □

EXAMPLE 1

Let $(H, \circ)$ be the Abelian monoid generated by $T(\Sigma \cup V)$.

Let $t_1 = x \circ a \circ x \circ a$ and $t_2 = g(y) \circ g(a) \circ z \circ z$.

We can find a "$\circ$"-endomorphism which equates t_1 and t_2 as follows:

- compute normal forms $h_1 = x^2 \circ a^2$ and $h_2 = g(y) \circ g(a) \circ z^2$;
- compute the equation E_{h_1, h_2}: $2x_1 + 2x_2 - x_3 - x_4 - 2x_5 = 0$;
- choose two solutions $s_1 = (10110)$, $s_2 = (11002) \in S_{h_1, h_2}$ and $k_1, k_2 \in T(\Sigma \cup V)$;
- define ϕ on x, a, $g(y)$, $g(a)$, z:

$$\phi(x) = k_1 \circ k_2, \qquad \phi(a) = k_2, \qquad \phi(g(y)) = k_1, \qquad \phi(g(a)) = k_1,$$
$$\phi(z) = k_2^2,$$

$\phi(h_1) = k_1^2 \circ k_2^4 = \phi(h_2)$. □

2. Endomorphism vs. Substitution

The next goal is to connect the notions of endomorphism and substitution. More precisely, we want to construct an endomorphism ϕ, as shown in section 1, and decide whether it "behaves" like a substitution.

TERMINOLOGY. Starting with this section, we shall use the following notation:

- $H - \{\varepsilon\} = T(\Sigma \cup V)$.
- $f \in \Sigma_{ac}$.
- $t_1 \circ t_2 = f(t_1, t_2)$ for $t_1, t_2 \in T(\Sigma \cup V)$;
  $h_1 = a_1^{p_1} \circ \ldots \circ a_m^{p_m}$
  $h_2 = a_{m+1}^{p_{m+1}} \circ \ldots \circ a_n^{p_n}$
  } as in section 1.
- ϕ is an endomorphism with $\phi(h_1) = \phi(h_2)$.
- $\{s_1, \ldots, s_r\} \in 2^{S_{h_1, h_2}}$ where S_{h_1, h_2} is the set of solutions defined in section 1.

In this section we want to see whether an "f"-endomorphism ϕ, which equates h_1 and h_2, can be regarded as a Σ-endomorphism, i.e. a morphism with respect to all operators. If so, ϕ corresponds to a substitution φ. To determine this we must address two questions:

- Does there exist a φ with $\phi(a_i) = \varphi(a_i)$ for all a_i?
- If yes, how can such a φ be constructed using $s_1, \ldots, s_r \in S_{h_1, h_2}$?

DEFINITION. Associated with each solution $s_j \in S_{h_1,h_2}$, we define the following quantities:

$$Q_j = \{i | 1 \leqslant i \leqslant n, \quad s_{ji} \geqslant 1, \quad a_i \notin V\},$$

$$q_j = \begin{cases} a_{\min(Q_j)}, & \text{if } Q_j \neq 0, \\ \text{a newly generated variable } z_j, & \text{otherwise.} \end{cases}$$

Given an endomorphism ϕ constructed via the method of the previous section satisfying

$$\forall\ 1 \leqslant i \leqslant n[\phi(a_i) = k_1^{s_{1i}} \circ \ldots \circ k_r^{s_{ri}}], \tag{1}$$

we wish to determine if there exists a substitution φ such that

$$\forall\ 1 \leqslant i \leqslant n[\phi(a_i) = \varphi(a_i)]. \tag{2}$$

Three direct consequences arise from (1) and (2):

$$\forall\ 1 \leqslant i \leqslant n\left[\sum_{j=1}^{r} s_{ji} \geqslant 1\right], \tag{3}$$

$$\forall\ 1 \leqslant i \leqslant n\left[a_i \notin V \Rightarrow \sum_{j=1}^{r} s_{ji} = 1\right], \tag{4}$$

$$\forall\ 1 \leqslant i_1, i_2 \leqslant n,\ 1 \leqslant j \leqslant r\ [i_1, i_2 \in Q_j \Rightarrow \exists \text{ substitution } \sigma\ [\sigma(a_{i_1}) = \sigma(a_{i_2})]]. \tag{5}$$

REMARK. The substitution σ is called a **unifier** for $\{h_1, h_2\}$.

PROOF OF (3) Assume the contrary. Then there exists an i with $\phi(a_i) = \varepsilon$, but a substitution φ cannot be ε for any $t \in T(\Sigma \cup V)$. □

PROOF OF (4). Assume there is an i with $\sum_{j=1}^{r} s_{ji} \geqslant 2$ and $a_i \notin V$. Then there exist terms t_1, t_2 such that $\phi(a_i) = t_1 \circ t_2 = f(t_1, t_2)$. From the normalisation of h_1 and h_2, a_i cannot be of the form $t_1' \circ t_2'$. However this contradicts (2), because φ, as a substitution, preserves the term structure. □

PROOF OF (5). Assume $i_1, i_2 \in Q_{j_1}$

$$\Rightarrow a_{i_1}, a_{i_2} \notin V \text{ by the definition of } Q_{j_1},$$

$$(4) \Rightarrow \sum_{j=1}^{r} s_{ji_1} = s_{j_1 i_1} = 1 \quad \text{and} \quad \sum_{j=1}^{r} s_{ji_2} = s_{j_1 i_2} = 1,$$

$$(1) \Rightarrow \phi(a_{i_1}) = k_{j_1} \quad \text{and} \quad \phi(a_{i_2}) = k_{j_1},$$

$$(2) \Rightarrow \varphi(a_{i_1}) = \varphi(a_{i_2}). \quad \square$$

The conditions (3)–(5) are necessary but not sufficient for the existence of a φ. We now present an algorithm which constructs φ:

(1) Let $\delta = \{a_i \leftarrow q_1^{s_{1i}} \circ \ldots \circ q_r^{s_{ri}} | a_i \in V\}$ and σ be the fixpoint of δ (if one exists). This means that if we can find an n, such that $\delta^n = \delta^{n+1}$, then $\sigma = \delta^n$.
(2) Find a substitution τ which unifies $\{\sigma(a_i) | i \in Q_j\}$ for all $1 \leqslant j \leqslant r$.
(3) Let $\varphi = \tau\sigma$ (if τ and σ exist). □

EXAMPLE 2. Let $h_1 = x^2 \circ a^2$, $h_2 = g(y) \circ g(a) \circ z^2$, $s_1 = (10110)$ and $s_2 = (11002)$, as in example 1. Then we construct $Q_1 = \{3, 4\}$, $q_1 = g(y)$, $Q_2 = \{2\}$ and $q_2 = a$. Obviously, s_1 and s_2 satisfy (3)–(5). We construct $\sigma = \delta = \{x \leftarrow g(y) \circ a,\ y \leftarrow a^2\}$. Then τ has to unify $\{a\}$ and $\{g(y), g(a)\}$. Finally, we get $\varphi = \{x \leftarrow g(a) \circ a,\ y \leftarrow a,\ z \leftarrow a^2\}$, with $\varphi(h_1) = g(a)^2 \circ a^4 = \varphi(h_2)$.

LEMMA 2. If the values k_j in (1) are defined as $k_j = \varphi(q_j)$ for $1 \leqslant j \leqslant r$, then (2) holds.

PROOF. Letting $a_i \in V$,

$$\varphi(a_i) = \tau\sigma(a_i) = \tau(\sigma(q_1)^{s_{1i}} \circ \ldots \circ \sigma(q_r)^{s_{ri}}) \quad (\sigma \text{ is a fixpoint of } \delta)$$
$$= \tau\sigma(q_1)^{s_{1i}} \circ \ldots \circ \tau\sigma(q_r)^{s_{ri}} = k_1^{s_{1i}} \circ \ldots \circ k_r^{s_{ri}} = \phi(a_i).$$

Letting $a_i \notin V$,

there exists exactly one $1 \leqslant j \leqslant r$ with $s_{ji} = 1$,
τ unifies $\{\sigma(a_i) | i \in Q_j\} \Rightarrow \varphi$ unifies $\{a_i | i \in Q_j\}$ $(\varphi = \tau\sigma)$,
$\Rightarrow \varphi(a_i) = \varphi(q_j) = k_j = \phi(a_i)$. □

COROLLARY. The substitution φ is a unifier for h_1 and h_2.

3. A Complete Set of Unifiers

DEFINITION. We use the following definitions (with some minor simplifications) from the survey by Huet & Oppen (1980).

- idempotent substitutions:

  δ is *idempotent* if and only if $\forall\ v \in V\ [\delta(v) = \delta\delta(v)]$.

- a *subsumption preorder* on substitutions:

  $\delta_1 \prec_X \delta_2$ iff $\exists\ \lambda\ \forall\ x \in X\ [\lambda\delta_1(x) = \delta_2(x)]$ (X is a set of variables).

- a *complete set of unifers* Ω for h_1 and h_2:

  $\forall\ \varphi \in \Omega\ [\varphi(h_1) = \varphi(h_2)$ and φ is idempotent],
  $\psi(h_1) = \psi(h_2) \Rightarrow \exists\ \varphi \in \Omega\ [\varphi \prec_{Vars(h_1, h_2)} \psi]$.

In this section we present an algorithm to construct a complete set of unifiers, and prove its correctness. The conditions (3)–(5) are used both to decrease the number of basic solutions that must be considered and to avoid repeating the computation of a particular unifier. This is crucial to make the algorithm efficient.

ALGORITHM. $\Omega \leftarrow AC\text{-}UNIFY(h_1, h_2)$

AC-UNIFY creates a complete set of unifiers for h_1 and h_2.

(1) Compute B_{h_1, h_2}, then regard all subsets B of B_{h_1, h_2}, for which (3)–(5) hold.
(2) Let $B = \{b_1, \ldots, b_r\}$, $b_j = (b_{j1}, \ldots, b_{jn})$, and $I = \{1 \leqslant i \leqslant n | a_i \in V\}$.
(I describes the set of all variable arguments.) Define Q_j and q_j as

$$Q_j = \{1 \leqslant i \leqslant n | b_{ji} = 1,\ i \notin I\},$$

$$q_j = \begin{cases} a_{\min(Q_j)}, & \text{if } Q_j \neq \emptyset, \\ z_j \text{ (a new variable)} & \text{otherwise,} \end{cases}$$

for $1 \leqslant j \leqslant r$.

(3) Construct $\delta^{(i)} = \{a_i \leftarrow q_1^{b_{1i}} \circ \ldots \circ q_r^{b_{ri}}\}$ for $i \in I$, and define δ as the union of all $\delta^{(i)}$. A fixpoint σ of δ exists if and only if the following sequence $i_1, \ldots, i_d$ exists with

- $I = \{i_1, \ldots, i_d\}$.
- $\forall\ 1 \leqslant e \leqslant d$ [no $v \in \{a_{i_e}, a_{i_{e+1}}, \ldots, a_{i_d}\}$ occurs in $\delta^{(i_e)}(a_{i_e})$].

Then $\sigma = \delta^{(i_1)} \ldots \delta^{(i_d)} = \{a_{i_e} \leftarrow \delta^{(i_1)} \ldots \delta^{(i_e)}(a_{i_e}) | 1 \leqslant e \leqslant d\}$ is this fixpoint.

(4) Use *UNIFY* to compute a complete set of unifiers Ω' for all sets $\{\sigma(a_i) | i \in Q_j\}$, and let $\Omega = \Omega \cup \{\tau\sigma | \tau \in \Omega'\}$. □

$\Omega \leftarrow UNIFY(t_1, t_2)$

Unification algorithm by Robinson (1965) which is extended for ac-terms. Ω is a complete set of unifiers for t_1 and t_2.

(1) If $t_1 = t_2$:
let $\Omega = \{\{\}\}$.

(2) If t_1 is a variable which does not occur in t_2:
let $\Omega = \{\{t_1 \leftarrow t_2\}\}$.

(3) If t_2 is a variable which does not occur in t_1:
let $\Omega = \{\{t_2 \leftarrow t_1\}\}$.

(4) If $t_1 = g(r_1, \ldots, r_k)$, $t_2 = g(s_1, \ldots, s_k)$, $g \notin \Sigma_{ac}$:
let $\Omega_1 = \{\{\}\}$;
for $1 \leqslant i \leqslant k$; let $\Omega_{i+1} = \{\varphi_1\varphi_2 | \varphi_2 \in \Omega_i,\ \varphi_1 \in UNIFY(\varphi_2(r_i), \varphi_2(s_i))\}$
let $\Omega = \Omega_{k+1}$.

(5) If $t_1 = f(r_1, \ldots, r_{k_1})$, $t_2 = f(s_1, \ldots, s_{k_2})$, $f \in \Sigma_{ac}$:
let $\Omega = AC\text{-}UNIFY(t_1, t_2)$.

(6) In all other cases:
let $\Omega = \emptyset$ (t_1 and t_2 are not unifiable). □

EXAMPLE 3. Given $h_1 = x^2 \circ a^2$ and $h_2 = g(y) \circ g(a) \circ z^2$ as in the previous examples. We compute eight basic solutions: $b_1 = (10110)$, $b_2 = (10200)$, $b_3 = (10020)$, $b_4 = (10001)$, $b_5 = (01110)$, $b_6 = (01200)$, $b_7 = (01020)$ and $b_8 = (01001)$. The solutions b_2, b_3, b_6 and b_7 cannot be part of any combination of basic solutions, because this would violate (4). Furthermore, b_5 violates (5). So only $\{b_1, b_8\}$ and $\{b_1, b_4, b_8\}$ satisfy (3)–(5). Note that of the 256 possible sets of basic solutions, there are 161 satisfying (3), of which 3 satisfy (4), and only 2 of which satisfy (5). The complete set of unifiers for h_1 and h_2 is shown in the appendix. □

THEOREM. $UNIFY(t_1, t_2)$ is a complete set of unifiers for t_1 and t_2.

First we have to prove that *UNIFY* terminates for any t_1 and t_2. The Noetherian (or well-founded) ordering we use was introduced by Fages (1984), who gives a detailed proof of the Stickel algorithm.

DEFINITION.

- A term t' is a *subterm* of term t if and only if
  $t' \neq t$ and $\exists$ term t'', occurrence $o[t = t'', t' = t''/o]$,
  ("=" denotes the congruence defined by all $f \in \Sigma_{ac}$).
  Example: $f(y, f(x, z))$ is a subterm of $g(f(x, f(y, z)))$, if $f \in \Sigma_{ac}$, $g \notin \Sigma_{ac}$.

- We are interested in a particular kind of subterms:
  A subterm t' of t is an *admissible subterm* if and only if:
  1. $t' = g(r_1, \ldots, r_m)$, $g \notin \Sigma_{ac}$
     or
  2. $t' = f(s_1, \ldots, s_n)$, $f \in \Sigma_{ac}$ and
     $\exists$ subterm $g(r_1, \ldots, r_m)$ of t, $g \neq f$, $\exists\ 1 \leqslant i \leqslant m$ with $t' = r_i$.

  Example: $f(x_1, x_2)$ is not an admissible subterm of $f(f(x_1, x_2), x_3)$, if $f \in \Sigma_{ac}$.
  By $AS(t)$ we denote the set of all admissible subterms of t.
- Let v be a variable, t_1, t_2 two terms.
  $Op(v, t_1, t_2) = \{g \in \Sigma | \text{function symbol } g \text{ occurs with argument } v \text{ in } t_1 \text{ or } t_2\}$.
  Example: $Op(x, f(g(x, y), x), f(x)) = \{f, g\}$.
- We define the *weight* of two terms t_1 and t_2 as the pair (α, β), where

$$\alpha = \#\{x \in V | \# Op(x, t_1, t_2) \geqslant 2\},$$
$$\beta = \#(AS(t_1) \cup AS(t_2)).$$

  Example: weight $(f(x), g(f(y), x)) = (1, 1)$, if $f \in \Sigma_{ac}$, $g \notin \Sigma_{ac}$:

$$\# Op(x, f(x), g(f(y), x)) = 2, \qquad \# Op(y, f(x), g(f(y), x)) = 1,$$
$$AS(f(x)) = \emptyset, \qquad AS(g(f(y), x)) = \{f(y)\}.$$

- Let s_1, s_2, t_1, t_2 be terms with $weight(s_1, t_1) = (\alpha_1, \beta_1)$ and $weight(s_2, t_2) = (\alpha_2, \beta_2)$. $(\alpha_1, \beta_1) < (\alpha_2, \beta_2)$ if and only if $\alpha_1 < \alpha_2$ or ($\alpha_1 = \alpha_2$ and $\beta_1 < \beta_2$). This defines a total Noetherian ordering on all pairs of terms.

LEMMA 3. Let s_1, s_2, t_1, t_2 be terms with: $t_1, t_2 \in AS(s_1) \cup AS(s_2) \cup \{s_1, s_2\}$,

$$t_1 = f(t_{11}, \ldots, t_{1m}), \qquad t_2 = f(t_{21}, \ldots, t_{2n}), \quad f \in \Sigma_{ac}.$$

Then σ as constructed by $AC\text{-}UNIFY(t_1, t_2)$ does not increase $weight(s_1, s_2)$.

PROOF. Let $a_i \leftarrow t \in \sigma$:

- t is a new variable:
  $\#\{x \in V | \# Op(x, s_1, s_2) \geqslant 2\}$ does not increase,
  $\Rightarrow weight(s_1, s_2)$ does not increase.
- $t = f(r_1, \ldots, r_p)$:
  $\# Op(a_i, s_1, s_2) \geqslant 2 \Rightarrow weight(s_1, s_2)$ decreases,
  $Op(a_i, s_1, s_2) = \{f\} \Rightarrow t$ is not introduced as a new admissible subterm
  $\Rightarrow weight(s_1, s_2)$ does not increase.

LEMMA 4. Let s_1, s_2, t_1, t_2 be terms with $t_1 \in AS(s_1) \cup AS(s_2)$, $\varphi \in UNIFY(t_1, t_2)$. Then $weight(\varphi(s_1), \varphi(s_2)) \leqslant weight(s_1, s_2)$. (**)

PROOF by induction over $weight(t_1, t_2)$.

1. $weight(t_1, t_2) = (0, 0)$:
   - $t_1 = g(t_{11}, \ldots, t_{1n})$, $t_2 = g(t_{21}, \ldots, t_{2n})$, $g \notin \Sigma_{ac}$:
     All arguments of t_1 and t_2 are variables, so we know for any $v_1 \leftarrow v_2 \in \varphi$:
     $v_1, v_2 \in V$ and $g \in Op(v_1, s_1, s_2) \cap Op(v_2, s_1, s_2)$,
     $\Rightarrow v_1 \leftarrow v_2$ does not increase $weight(s_1, s_2)$.

- $t_1 = f(t_{11}, \ldots, t_{1m})$, $t_2 = f(t_{21}, \ldots, t_{2n})$, $f \in \Sigma_{ac}$:
  (**) follows from Lemma 3.

2. $weight(t_1, t_2) = (\alpha, \beta)$:
Assume no $\varphi' \in UNIFY(t_1', t_2')$ increases $weight(s_1', s_2')$ for terms s_1', s_2', t_1', t_2' with $t_1', t_2' \in AS(s_1') \cup AS(s_2')$, $weight(t_1', t_2') < weight(t_1, t_2)$.

- $t_1 = g(t_{11}, \ldots, t_{1n})$, $t_2 = g(t_{21}, \ldots, t_{2n})$, $g \notin \Sigma_{ac}$:
  Let $\varphi' \in UNIFY(t_{1i}, t_{2i})$ for $1 \leqslant i \leqslant n$.
  $t_{1i} \in V$ (or $t_{2i} \in V$):
  $\varphi' = \{t_{1i} \leftarrow t_{2i}\} \Rightarrow weight(s_1, s_2)$ does not increase.
  $t_{1i} \in AS(t_1)$, $t_{2i} \in AS(t_2) \Rightarrow$ (**) follows from the induction hypothesis.
- $t_1 = f(t_{11}, \ldots, t_{1m})$, $t_2 = f(t_{21}, \ldots, t_{2n})$, $f \in \Sigma_{ac}$:
  From Lemma 3 we know that σ does not increase $weight(s_1, s_2)$, so we have to show that unification of a pair $(\sigma(a_{i_1}), \sigma(a_{i_2}))$ with $a_{i_1}, a_{i_2} \notin V$ does not increase the weight:
  $a_{i_1}, a_{i_2} \notin V \Rightarrow a_{i_1}, a_{i_2} \in AS(t_1) \cup AS(t_2) \Rightarrow a_{i_1}, a_{i_2} \in AS(s_1) \cup AS(s_2)$
  $\Rightarrow \sigma(a_{i_1}), \sigma(a_{i_2}) \in AS(\sigma(s_1)) \cup AS(\sigma(s_2))$.
  Now (**) follows from the induction hypothesis. □

LEMMA 5 (termination). $UNIFY(t_1, t_2)$ terminates for any t_1 and t_2.

PROOF. We have to show that for every recursive call of *UNIFY* either one of the arguments is a variable (in which case termination is obvious) or the weight of the arguments decreases:

- $t_1 = g(t_{11}, \ldots, t_{1n})$, $t_2 = g(t_{21}, \ldots, t_{2n})$, $g \notin \Sigma_{ac}$:
  Lemma 4 shows that unification of a pair (t_{1i}, t_{2i}) of nonvariable arguments does not increase $weight(t_1, t_2)$. $weight(t_{1i}, t_{2i}) < weight(t_1, t_2)$, because t_{1i} and t_{2i} are admissible subterms.
- $t_1 = f(t_{11}, \ldots, t_{1m})$, $t_2 = f(t_{21}, \ldots, t_{2n})$, $f \in \Sigma_{ac}$:
  σ does not increase $weight(t_1, t_2)$ (Lemma 3), nor does unification of a pair $(\sigma(a_{i_1})\, \sigma(a_{i_2}))$ (Lemma 4). As shown in Lemma 4, $\sigma(a_{i_1})$ and $\sigma(a_{i_2})$ are admissible subterms of $\sigma(t_1)$ or $\sigma(t_2)$, so
  $$weight(\sigma(a_{i_1}), \sigma(a_{i_2})) < weight(\sigma(t_1), \sigma(t_2)).$$ □

REMARK. Implicitly, all the following proofs are inductions over the weight of the arguments to *UNIFY*.

LEMMA 6 (correctness). All substitutions $\varphi \in UNIFY(h_1, h_2)$ are unifiers for h_1 and h_2.

PROOF. This follows from Lemma 2 for the *ac*-case. Obviously, step (1)–(4) of *UNIFY* compute unifiers (also see Robinson (1965)).

REMARK. We can regard every unifier ψ for h_1 and h_2 as an endomorphism on $(H, \circ)$. ψ defines $b_1, \ldots, b_r \in B_{h_1, h_2}$ and $k_1, \ldots, k_r \in T(\Sigma \cup V)$ with
$$\forall\, 1 \leqslant i \leqslant n\ [\psi(a_i) = k_1^{b_{1i}} \circ \ldots \circ k_r^{b_{ri}}].$$

LEMMA 7. Let ψ be a unifier for h_1 and h_2, which defines $b_1, \ldots, b_r$ and $k_1, \ldots, k_r$. Then *AC-UNIFY* computes a σ with $\sigma \prec_{Vars(h_1, h_2)} \psi$.

PROOF. Without loss of generality we may assume that ψ is idempotent. Since ψ is a unifier for h_1 and h_2, (3)–(5) hold for $b_1, \ldots, b_r$, and there exists a sequence $i_1, \ldots, i_d$. Let $\lambda = \{q_j \leftarrow k_j | q_j \text{ is a new variable}\}$. Now we prove by induction $\forall\, i \in I[\psi\lambda\sigma(a_i) = \psi(a_i)]$!

$i = i_1$:
$\psi\lambda\sigma(a_i) = \psi\lambda\delta^{(i)}(a_i) = \psi\lambda(q_1^{b_{1i}} \circ \ldots \circ q_r^{b_{ri}})$,
q_j is a new variable $\Rightarrow \psi\lambda(q_j) = \psi(k_j) = k_j$,
q_j is a nonvariable $a_i \Rightarrow \psi\lambda(q_j) = \psi(q_j) = k_j$ by definition of q_j, b_j and k_j,
$\Rightarrow \psi\lambda\sigma(a_i) = k_1^{b_{1i}} \circ \ldots \circ k_r^{b_{ri}} = \psi(a_i)$.

$i = i_e$, $2 \leqslant e \leqslant d$:
$\psi\lambda\sigma(a_i) = \psi\lambda\sigma\delta^{(i)}(a_i)$ (see *AC-UNIFY* step (3)) $= \psi\lambda\sigma(q_1^{b_{1i}} \circ \ldots \circ q_r^{b_{ri}})$,
q_j is a new variable $\Rightarrow \psi\lambda\sigma(q_j) = \psi\lambda(q_j) = \psi(k_j) = k_j$,
q_j is a nonvariable a_i, $v \in Vars(q_j)$:
$v \in \{a_{i_1}, \ldots, a_{i_{e-1}}\} \Rightarrow \psi\lambda\sigma(v) = \psi(v)$ by induction hypothesis,
$v \notin I$ (v cannot be $a_{i_e}, \ldots, a_{i_d}$) $\Rightarrow \psi\lambda\sigma(v) = \psi\lambda(v) = \psi(v)$,
$\Rightarrow \psi\lambda\sigma(q_j) = \psi(q_j) = k_j$,
$\Rightarrow \psi\lambda\sigma(a_i) = k_1^{b_{1i}} \circ \ldots \circ k_r^{b_{ri}} = \psi(a_i)$.

Now we know $\sigma \prec_X \psi$, where X is the set $\{a_i | i \in I\}$. σ is defined on X, so $\sigma \prec_{Vars(h_1, h_2)} \psi$. □

LEMMA 8 (completeness). $UNIFY(t_1, t_2)$ is a complete set of unifiers for t_1 and t_2.

Proof. Let ψ be any unifier for t_1 and t_2. We have to show that there exists a $\varphi \in UNIFY(t_1, t_2)$ with $\varphi \prec_{Vars(t_1, t_2)} \psi$.

- $t_1 \in V$ (or $t_2 \in V$):
  $\varphi = \{t_1 \leftarrow t_2\} \Rightarrow \forall\, v \in Vars(t_1, t_2)\ [\psi\varphi(v) = \psi(v)]$.
- $t_1 = g(r_1, \ldots, r_k)$, $t_2 = g(s_1, \ldots, s_k)$, $g \notin \Sigma_{ac}$:
  We prove by induction $\forall\, 1 \leqslant i \leqslant r+1\ \exists\, \varphi \in \Omega_i\ [\varphi \prec_{Vars(t_1, t_2)} \psi]$!
  - $\varphi \in \Omega_1 \Rightarrow \varphi = \{\} \Rightarrow \varphi \prec_{Vars(t_1, t_2)} \psi$.
  - Assume there exists a $\varphi_2 \in \Omega_i$, $1 \leqslant i \leqslant r$, with $\varphi_2 \prec_{Vars(t_1, t_2)} \psi$.
    Let $X_1 = Vars(t_1, t_2)$, $X_2 = Vars(\varphi_2(r_i), \varphi_2(s_i))$.
    $\exists\, \lambda_2\ \forall\, v \in X_1\ [\lambda_2\varphi_2(v) = \psi(v)]$ and ψ is a unifier for r_i and s_i,
    $\Rightarrow \lambda_2$ is a unifier for $\varphi_2(r_i)$ and $\varphi_2(s_i)$,
    $\exists\, \varphi_1 \in UNIFY(\varphi_2(r_i), \varphi_2(s_i))\ [\varphi_1 \prec_{Vars(\varphi_2(r_i), \varphi_2(s_i))} \lambda_2)]$,
    $\Rightarrow \exists\, \lambda_1\ \forall\, v \in X_2\ [\lambda_1\varphi_1(v) = \lambda_2(v)]$,
    $\Rightarrow \forall\, v \in X_1\ [\lambda_1\varphi_1\varphi_2(v) = \lambda_2\varphi_2(v) = \psi(v)] \Rightarrow \varphi_1\varphi_2 \prec_{X_1} \psi$,
  $\Rightarrow \exists\, \varphi\ \in \Omega\ [\varphi \prec_{Vars(t_1, t_2)} \psi]$.
- t_1 and t_2 are *ac*-terms:
  By Lemma 7 we know that *AC-UNIFY* computes a σ with $\sigma \prec_{Vars(t_1, t_2)} \psi$. Similar to the previous case we can show that the recursive calls of *UNIFY* compute a τ with $\tau\sigma \prec_{Vars(t_1, t_2)} \psi$. □

DEFINITION. Ω is a *minimal set of unifiers* for h_1 and h_2 if and only if:

- $\forall\, \varphi \in \Omega\ [\varphi(h_1) = \varphi(h_2)$ and φ is idempotent$]$.
- $\forall\, \varphi_1, \varphi_2 \in \Omega\ [\varphi_1 \prec_{Vars(h_1, h_2)} \varphi_2 \Rightarrow \varphi_1 = \varphi_2]$.

UNIFY computes a complete set of unifiers. Earlier this procedure was believed to produce a minimal set (Fortenbacher, 1985), but the last unification in appendix B is a counterexample: Let φ_1 be the first, φ_2 the fifth unifier computed, and let $\lambda = \{W \leftarrow +(V4, V3), Z \leftarrow +(V3, V1\}$. Then $\forall\, v \in \{W, X, Y, Z\}[\lambda\varphi_2(v) = \varphi_1(v)]$.

4. An Efficient Algorithm

Our next goal is to find an efficient implementation for *AC-UNIFY*. First we have to compute the finite set of basic solutions of a homogeneous diophantine equation. There are algorithms proposed by Huet (1978), Fortenbacher (1983), Lankford (1985) and Büttner (1985). The first two are discussed and compared in Guckenbiehl & Herold (1985).

Given a set of solutions, how can we find all subsets thereof which can be used to compute a unifier? All subsets deserving attention must satisfy (3) for correctness. Without losing completeness, subsets can also be constrained to satisfy (4), and a basic solution which violates (5) need not be part of any subset. This leads to the following procedure to compute all promising subsets:

(1) Eliminate every basic solution which cannot be a member of any subset. This is the case if $b_{jl} \geqslant 2$ for an $a_l \notin V$, which violates (4), or if the set $\{a_i | i \in Q_j\}$ is not unifiable ((5)).
(2) With regard to (1) find all subsets which satisfy (3) and (4). □

Stickel (1981) discusses to apply the constraints (4) and (5) during the generation of unifiers in the variable-only case rather than afterward.

Now we address the problem that the set $\{a_i | i \in Q_j\}$ has to be unified for each subset of solutions containing b_j. Clearly, an efficient implementation has to obviate the need for repeatedly computing these unifiers. But how thoroughly should the testing for (5) be done? While implementing the algorithm (see appendix B), the author considered three alternatives:

P1: Make a plausibility check to test for (5): Disregard all pairs with the same *ac*-operator and look for a "disagreement pair", this is a pair of subterms that has two different operators. Unification is impossible if such a disagreement pair exists.
P2: Unify all $\{a_i | i \in Q_j\}$, but postpone pairs with the same *ac*-operator. This computes a substitution and a set of unsolved unification problems for every b_j. Hullot (1979) uses a similar technique when he regards subsets of solutions.
P3: Compute a complete set of unifiers for every $\{a_i | i \in Q_j\}$.

The number of unsuitable solutions detected grows from P1 to P3, which makes the number of subsets shrink. A pair $g(x, x), g(a, b)$ passes P1, whereas a pair $f(x, x), f(a, b)$ even passes P2, but not P3 ($f \in \Sigma_{ac}$, $g \notin \Sigma_{ac}$). The number of unifications performed decreases from P1 to P3 if b_j satisfies (5). P1 repeatedly performs all unifications, P2 all *ac*-unifications. But is P3 the most efficient alternative? This depends heavily on the particular implementation, and especially on the underlying data structure. Robinson (1971) showed that the substitution of terms can be performed very cheaply by updating pointers. But in this context saving a substitution means to save an environment, which involves a lot of copying. This is especially significant if we want to unify two *ac*-terms with many variable arguments (see the last example of appendix B), whereas there is at most one unifier for two terms with no *ac*-operator. This was the reason for the author's decision to implement P2.

References

Bütter, W. (1985). *Unification in the datastructure multisets.* Report, Siemens AG, Corporate Laboratories for Information Technology, München, 1985 (to appear in *J. Autom. Reas*).

Collins, G. E. (1980). Aldes and SAC-2 now available. *SIGSAM Bull.* 10/2.

Fages, F. (1984). Associative-commutative unification. *Proc. of the 7th Int. Conf. on Automated Deduction.* Berlin: Springer.

Fortenbacher, A. (1983). *Algebraische Unifikation.* Diplomarbeit, Institut für Informatik I, Universität Karlsruhe.

Fortenbacher, A. (1985). Unification under associativity and commutativity. *Proc. of the 1st Int. Conf. on Rewriting Techniques and Applications.* Berlin: Springer.

Guckenbiehl, T., Herold, A. (1985). *Solving linear diophantine equations.* MEMO SEKI-III-KL, Universität Kaiserslautern.

Herold, A., Siekmann, J. H. (1985). *Unification in abelian semigroups.* MEMO SEKI-III-KL, Universität Kaiserslautern.

Hullot, J. M. (1979). Associative-commutative pattern matching. *5th Int. Joint Conf. on Artificial Intelligence,* Tokyo.

Huet, G., Oppen, D. C. (1980). Equations and rewrite rules: a survey. In: (Book, R., ed) *Formal languages: perspectives and open problems.* New York: Academic Press.

Huet, G. (1978). An algorithm to generate the basis of solutions to homogeneous linear diophantine equations. *Inf. Proc. Lett.* 7/3.

Küchlin, W. W. (1982). *An implementation and investigation of the Knuth–Bendix completion procedure.* Interner Bericht, Institut für Informatik I, Universität Karlsruhe.

Lankford, D. *A new non-negative integer basis algorithm for linear homogeneous equations with integer coefficients* (unpublished).

Loos, R. G. K. (1976). The algorithm description language ALDES (report). *SIGSAM Bull.* 10/1.

Robinson, J. A. (1965). A machine-oriented logic based on the resolution principle. *J. Assoc. Comp. Mach.* **12.**

Robinson, J. A. (1971). Computational logic. The unification computation. In: (Michie & Meltzer, eds) *Machine Intelligence,* 6. Edinburgh: Edinburgh University Press.

Siekmann, J. H., Livesey, M. (1976). *Unification of $A+C$-terms (bags) and $A+C+I$-terms (sets).* Interner Bericht 3/76, Institut für Informatik, Universität Karlsruhe.

Stickel, M. E. (1975). A complete unification algorithm for associative-commutative functions. *Proc. 4th IJCAI,* Tblisi, USSR.

Stickel, M. E. (1981). A unification algorithm for associative-commutative functions. *J. Assoc. Comp. Mach.* **28.**

Appendix A: Matching

Matching is one-way unification where unification is permitted in only one of the terms. A substitution φ is a match of terms t_1 and t_2 if and only if $\varphi(t_1) = t_2$. In a term rewrite system, matching is the most frequent operation performed.

The following presents a straightforward method to implement *MATCH* using *UNIFY*. Supposedly there are more efficient realizations for term rewrite systems, for two reasons: First, there are better methods to compute a complete set of matches than that of solving diophantine equations (see Hullot, 1979). Second, in order to reduce a term, we need one match from the left side of a rule to a subterm, not a complete set of matches.

DEFINITION. A set of substitutions Ω is a complete set of matches for t_1 and t_2 if and only if

$$\forall\ \varphi \in \Omega\ [\varphi(t_1) = t_2],$$

$$\psi(t_1) = t_2 \Rightarrow \exists\ \varphi \in \Omega\ [\varphi \prec_{Vars(t_1)} \psi].$$

A way to implement an algorithm *MATCH* is the following:

(1) Replace all variables in t_2 by new constants, and call the new term t.
(2) Call $UNIFY(t_1, t)$.
(3) Now replace the new constants by the original variables in all unifiers. □

LEMMA 10 (correctness). All substitutions $\varphi \in MATCH(t_1, t_2)$ are matches of t_1 and t_2.

PROOF. Let $\varphi \in MATCH(t_1, t_2)$ and $\sigma \in UNIFY(t_1, t)$ the corresponding unifier. t does not contain any variables, so $\sigma(t) = t$ and $\sigma(t_1) = \sigma(t) = t$. $\varphi(t_1) = \sigma(t_1)$ with all new constants replaced by their original variables, and therefore $\varphi(t_1) = t_2$. □

LEMMA 11 (completeness). $MATCH(t_1, t_2)$ is a complete set of matches for t_1 and t_2.

PROOF. Let ξ be a match for t_1 and t_2, and ψ the substitution obtained from ξ by variable-constant replacement. ψ is a unifier for t_1 and t_2, so we can find a $\varphi \in UNIFY(t_1, t_2)$ with $\varphi \prec_{Vars(t_1, t_2)} \psi$. This implies the existence of a $\tau \in MATCH(t_1, t_2)$ with $\tau \prec_{Vars(t_1, t_2)} \xi$. □

Appendix B: Unification in SAC-2

The following is the image of a computer session run on an IBM 3081 under CMS at the IBM Watson Research Center, Yorktown Heights. The algorithm, which is written in ALDES (Loos, 1976), uses the computer algebra system SAC-2 (Collins, 1980) as well as a package for data type completion by Küchlin (1982). For more information see Fortenbacher (1983).

```
THE FOLLOWING DATA TYPE WAS READ

     TYPE T.
     CONSTS A,B,C - T
     VARS W,X,Y,Z - T
     OPS F(T,T) - T . G(T) - T . +(T,T) - T
     AC F,+
     END

UNIFICATION OF F(A,X) AND F(B,Y)
     Y - F(A,V1) . X - F(B,V1)
     Y - A . X - B
IN 5 MS

UNIFICATION OF +(X,X) AND +(Y,Z)
     Z - +(V3,V3,V2) . Y - +(V2,V1,V1) . X - +(V3,V2,V1)
     Y - +(Z,V1,V1) . X - +(Z,V1)
     Z - +(V3,V3) . Y - +(V1,V1) . X - +(V3,V1)
     Z - +(V3,V3,Y) . X - +(V3,Y)
     Z - X . Y - X
IN 12 MS

UNIFICATION OF F(X,X,Y,A,C) AND F(B,B,Z,C)
     Z - F(A,V2,X,X) . Y - F(V2,B,B)
     Z - F(A,X,X) . Y - F(B,B)
     Z - F(A,Y,V1,V1) . X - F(B,V1)
     Z - F(A,Y) . X - B
IN 12 MS

UNIFICATION OF F(+(X,A),+(Y,A),C,C) AND F(+(Z,Z,Z),W)
     W - F(+(X,A),C,C) . Z - +(A,V1) . Y - +(A,A,V1,V1,V1)
** 1934 CELLS, 7 MS
     W - F(+(X,A),C,C) . Z - A . Y - +(A,A)
     W - F(+(Y,A),C,C) . Z - +(A,V2) . X - +(A,A,V2,V2,V2)
     W - F(+(Y,A),C,C) . Z - A . X - +(A,A)
IN 30 MS

UNIFICATION OF F(+(X,A),+(Y,A),+(Z,A)) AND F(+(W,W,W),Z,Z)
     W - +(A,V1) . Z - +(+(A,V1,V1,V1),A) . Y - +(A,V1,V1,V1) . X - +(A,V1,V1,V1)
     W - A . Z - +(A,A) . Y - A . X - A
IN 19 MS

UNIFICATION OF F(X,A,X,A) AND F(G(A),G(Y),Z,Z)
     Z - F(A,V1) . Y - A . X - F(V1,G(A))
     Z - A . Y - A . X - G(A)
IN 9 MS
```

```
UNIFICATION OF +(W,+(X,G(X))) AND +(Y,+(Z,G(Z)))
     Z - +(V3,V1) , Y - +(V4,V3) , X - +(V3,V1) , W - +(V4,V3)
     Y - +(V4,Z) , X - Z , W - +(V4,Z)
     Z - +(Y,V1) , X - +(Y,V1) , W - Y
     Z - W , Y - W , X - W
     Y - W , X - Z
     Z - +(V2,V1) , Y - +(V4,V3,G(+(V3,V1,G(+(V2,V1))))) , X - +(V3,V1,G(+(V2,V1))), W -
          +(V4,V2)
     Y - +(W,V3,G(+(V3,Z,G(Z)))) , X - +(V3,Z,G(Z))
     Y - +(V4,V3,G(+(V3,G(Z)))) , X - +(V3,G(Z)) , W - +(V4,Z)
     Z - +(W,V1) , Y - +(V3,G(+(V3,V1,G(+(W,V1))))) , X - +(V3,V1,G(+(W,V1)))
     Z - W , Y - +(V3,G(+(V3,G(W)))) , X - +(V3,G(W))
     Z - +(V2,V1) , Y - +(V4,G(+V1,G(+V2,V1))))) , X - +(V1,G(+V2,V1))) , W - +(V4,V2)
     Y - +(W,G(+(Z,G(Z)))) , X - +(Z,G(Z))
     Y - +(V4,G(G(Z))) , X = G(Z) , W - +(V4,Z)
     Z - +(W,V1) , Y - G(+(V1,G(+(W,V1)))) , X - +(V1,G(+(W,V1)))
     Z - W , Y - G(G(W)) , X - G(W)
     Z - +(V2,V1,G(+(V3,V1))) , Y - +(V4,V3) , X - +(V3,V1) , W -
          (V4,V2,G(+(V2,V1,G(+V3,V1)))))
     Z - +(V1,G(+(V3,V1))) , Y - +(V4,V3) , X - +(V3,V1) , W -
          +(V4,G(+(V1,G(+(V3,V1)))))
** 1827 CELLS, 7 MS.
     Z - +(V2,G(X)) , Y - +(V4,X) , W - +(V4,V2,G(+(V2,G(X))))
     Z - G(X) , Y - +(V4,X) , W - +(V4,G(G(X)))
     Z - +(V2,V1,G(+(Y,V1))) , X - +(Y,V1) , W - +(V2,G(+V2,V1,G(+(Y,V1))))
     Z - +(V1,G(+(Y,V1))) , X - +(Y,V1) , W - G(+(V1,G(+(Y,V1))))
     Z - +(V2,G(X)) , Y - X , W - +(V2,G(+(V2,G(X))))
     Z - G(X) , Y - X , W - G(G(X))
     Z - +(V2,X,G(X)) , W - +(Y,V2,G(+(V2,X,G(X))))
     Z - +(X,G(X)) , W - +(Y,G(+(X,G(X))))
     Z - +(V2,V1) , Y - +(V4,V3,G(+(V3,V1))) , X - +(V3,V1) , W - +(V4,V2,G(+(V2,V1)))
     Y - +(V4,V3,G(+(V3,Z))) , X - +(V3,Z) , W - +(V4,G(Z))
     Y - +(V4,X,G(X)) , W - +(V4,Z,G(Z))
     Z - +(V2,V1) , Y - +(V3,G(+(V3,V1))) , X - +(V3,V1) , W - +(V2,G(+(V2,V1)))
     Y - +(V3,G(+(V3,Z))) , X - +(V3,Z) , W - G(Z)
     Y - +(X,G(X)) , W - +(Z,G(Z))
     Z - +(V2,X) , Y - +(V4,G(X)) , W - +(V4,V2,G(+(V2,X)))
     Z - X , Y - +(V4,G(X)) , W - +(V4,G(X))
     Z - +(V2,X) , Y - G(X) , W - +(V2,G(+(V2,X)))
     Z - X , Y - G(X) , W - G(X)
IN 108 MS
```